From Validation of Medical Devices towards Validation of Adaptive Cyber-Physical Systems

Tavčar, Jože; Duhovnik, Jože; Horváth, Imre

DOI
10.3233/JID190008

Publication date
2020

Document Version
Accepted author manuscript

Published in
Journal of Integrated Design and Process Science

Citation (APA)
Tavčar, J., Duhovnik, J., & Horváth, I. (2020). From Validation of Medical Devices towards Validation of Adaptive Cyber-Physical Systems. *Journal of Integrated Design and Process Science*, *23*(1), 37-59. https://doi.org/10.3233/JID190008

From Validation of Medical Devices towards Validation of Adaptive Cyber-Physical Systems

Jože Tavčar[a*], Jože Duhovnik[a], and Imre Horváth[b]

[a] *Faculty of Mechanical Engineering, University of Ljubljana, Ljubljana, Slovenia*
[b] *Faculty of Industrial Design Engineering, Delft University of Technology, Delft, Netherlands*

Abstract Conventionally, designing of a technical system is completed in the product development phase in terms of all details and aspects. This can be done if the expected behavior and the conditions of operation are known in advance and can be validated before the product is launched on the market. Validation is typically based on a predictive analysis or simulation of the designed operation. However, this contradicts with the case of smart systems, such as smart cyber-physical systems (CPSs), which self-manage their operation or at least a part of it. Being able to adapt at run-time and evolve over time, smart CPSs cannot be validated using conventional deterministic approaches. This is especially true for smart CPSs used as instrumentation in the medical field. This gave the stimulation of our background research, the results of which are concisely summarized and critically concluded in this paper. The literature has been found fairly narrow in terms of novel validation approaches for self-managing smart systems. The main finding is that the tasks of operational and behavioral validation should be shared among the system designers and the designed systems. Designers need prognostic approaches, while systems need to construct validation plans and execute them at run-time. This needs validation-specific functionalities and context dependent mechanisms such as run-time validation frameworks or meta-models, objective-sensitive self-monitoring mechanisms, self-constraining and self-supporting mechanisms, and other enablers. Extensive foundational research and system prototype testing are deemed to be indispensable. To make the first small step in this direction, this paper proposes a concept for validation of smart medical CPSs. This relies on the following hypothesis: If a system has freedom for self-adaptation, then it should also be equipped with self-control mechanism, meta-knowledge and a supervisory controller. It all together enables a purpose- and context dependent semantic reasoning about the operational and behavioral objectives. The paper suggests a number of topics for future research towards a run-time validation engine.

Keywords: Smart cyber-physical system, operational and behavioural validation, run-time validation, validation strategy, research issues

1. Introduction

1.1 Setting the stage for discussion

Each system needs to be verified and validated in some way before put into operation. Validation succeeds verification, which concerns the realization of the design objectives and the overall requirements. Certain aspects of validation can be considered during verification of designs, but design verification is not a substitute for validation. It is a prelude of it. Verification informs about if a system is improper, inconsistent and/or faulty and then it cannot be validated as appropriate for a given purpose. Validation ensures that the technical expectations of the users and the operational environment are met and that the intended uses can be fulfilled in a consistent manner (Katz, and Campbell, 2012). In other words, validation checks if a system will operate consistently according to a predetermined specification and fulfil the operational and behavioral requests of customers. Validation requires decision making on the operational qualities, identifying the possible influential factors, planning the tests, setting the

[*] Corresponding author. Email: joze.tavcar@lecad.fs.uni-lj.si Tel: (+386) 1 4771 415.

measurement parameters, and the evaluation of fitness for purpose. Likewise, the results of validation, these are included in a documented execution and evaluation plan.

The issue of assuring that complex systems, such as a smart cyber-physical system (CPS), will work properly under all circumstances, especially in safety critical situations, is often referred to as a challenge in the literature, but also in the practice. Validation is of high importance in the case of health/medical systems, unmanned vehicles, aircrafts and aerospace systems, smart manufacturing systems, etc. These systems typically change the objective and characteristics of their operation according to their state and the influence of the operational environment. This ability is referred to as adaptation. Contrary to their run-time adaptation, they should be kept under control, i.e. they should work safely and achieve optimal performance under varying circumstances. Existing mainstream model checking techniques and tools were not conceived for run-time usage; hence they hardly meet the constraints imposed by on-the-fly analysis in terms of execution time and memory usage (Filieri et al., 2016).

There has been a knowledge gap recognized between the traditional validation practices and the necessary validation approaches for smart systems, which feature novel behavioral and self-adaptation abilities. Due to the only partially foreseeable adaptation, the traditional, deterministic, and human-completed way of validation does not work in the case of smart CPSs. Furthermore, validation is also challenging not only due to the growing intelligence of the systems, but also due to the growing complexity of systems. System complexity implies an overwhelming number of states to be checked and this can often not be done in an acceptable time and cost frame. There is an intention to use smart CPSs more and more in safety critical applications. However, the traditional control that is usually based on the principle of 'human in the loop' needs to be replaced by some sort of 'control automatism' in these applications. Currently, the issue of concurrent validation and dynamic trust building are the main obstacles for moving in this direction. On the other hand, developing systems which have the ability of self-awareness, self-generation of operational constraints, and run-time self-validation is a promising way forward (Mitsch, and Platzer, 2014).

This paper starts out from an overview of the possibilities, limitations and open issues of conventional system validation procedures. The next subsection discusses the distinguishing characteristics of smart cyber-physical systems and sketches up the latest academic results and the industrial principles of the current validation practice. Section 2 analyses the validation process of traditional technical systems. Section 3 discusses the current approaches and implementation of validation of CPSs. Specifically, it focusses on the issues of run-time monitoring of complex systems, run-time monitoring of software operation, fault discovery and elimination, and considers platforms, smart components, and decentralized control as enablers. Based on the findings, it proposes a comprehensive approach that considers system verification as an essential prerequisite of behavioral validation. The proposal includes a checklist for system verification as well as methods for assessment of risk, checking the building blocks and sub-systems, and construction of a validation plan. Run-time validation model proposal is a specific topic. The execution process of validation is discussed, together with the issues of reporting, the corrective actions, and final decision making. The paper is focused on the approach "done by human and done in design phase". Based on these, the proposed approach can be operationalized in the industrial practice of validation of smart CPSs. Due to space limitation and ongoing research this paper only partly covers open issues related to "done by system and done in run-time phase".

1.2 Characteristics of smart cyber-physical systems

The first generation CPSs are able to self-adjust or self-tune their task execution to pre-set operational objectives during operation. The architecture and the default operation of these systems are defined in the design phase. These systems have conventional control mechanisms by means of which they can regulate parameters in a given range (Horváth et al., 2017). The main distinguishing characteristics of smart CPS are purpose-driven and context-dependent awareness building and reasoning that make architectural and functional adaptation and evolution possible - provided they are equipped with or can acquire the needed resources. Adaptation and evolution mean that smart CPSs change their features and configuration run-time, or even during the entire system life cycle, respectively, in a way that was not completely known in the design time (Horváth, and Gerritsen, 2012).

Quasi-smartly behaving CPSs establish the entry level of second generation CPSs and mean a transition step toward truly smarty behaving CPSs (Horváth et al., 2017). The main difference between them is in the level of self-management with regard to setting their operational objectives and building a control model. Quasi-smart CPSs has the ability to change the predefined control algorithms during the exploitation period (Fig. 1). However, the adaptation can happen without fundamentally changing the control model, or only inside a predefined parameter interval or an envelope, and with resources constrained because of safety or any other reasons determined in the design phase. The primary enabling asset of CPSs is cyber-physical computing (CPC), which is based on processing run-time obtained data, information and knowledge (Aßmann et al., 2014).

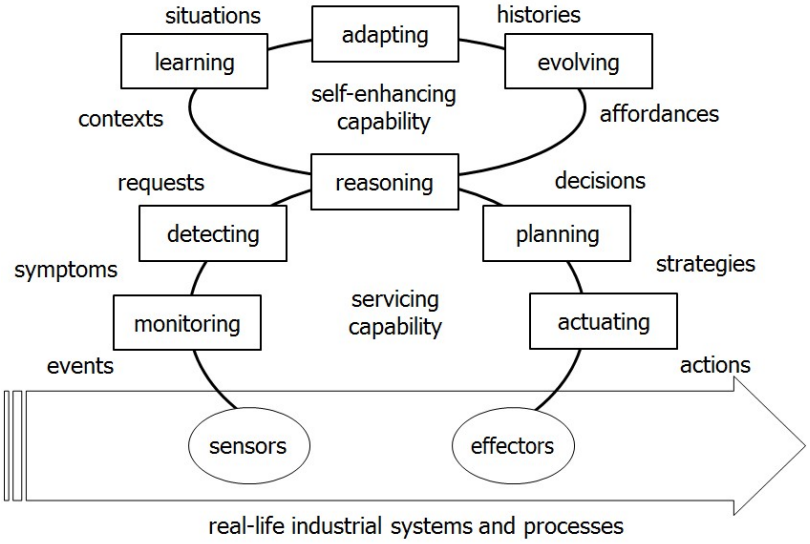

Fig. 1. Dual control cycles of S-CPSs

Smart CPSs are complex, partially autonomous systems, showing emergence, dynamics, non-linearity, and other behaviors not present in the elements (Marwedel, 2011). Truly smart CPSs represent the second generation of this family of systems, why cognizant CPSs belong to the third generation of CPSs. Depending on the nature and application context, smart CPSs may show both non-linear operation and emergent behavior in run-time that may be associated with their hardware, software, and cyberware constituents. From the point of view of system control, they implement two recurrent and intertwined cycles of computing, namely: (i): the basic control cycle, which includes 'sensing → monitoring → reasoning → actuating' activities, and (ii) the (self-)enhancement cycle, comprising 'reasoning → learning → adapting → evolving' activities (Fig. 1). This entails that multi-actor S-CPSs are not fully deterministic systems. They have some level of freedom in setting the objectives of system operations, as well as the capability of adapting themselves to varying tasks, situations, and contexts (Tavčar, and Horváth, 2018).

1.3 Limitations and open issues of current validation practices

It is widely known that formal theories, frameworks and methodologies of system level verification and validation are somewhat unconsolidated. There are three major reasons for it: (i) verification, but in particular validation, may have numerous objectives and aspects for testing, (ii) the indicators of validity strongly depend on the field of application that works against uniformity, and (iii) complexity and heterogeneity of systems make the standardization complicated. This is reflected by the majority of the papers we analyzed in our literature study. For instance, verification and validation of urban driving systems are an unsolved problem (Toben et al., 2012), even though significant progress has been made in recent years (Urmson et al., 2008). It is expected that research in unmanned vehicles will make an important contribution for civil applications if there will be fewer restrictions by the military (Finn, and Scheding, 2010). Current flight control systems are already based on a reasoning mechanism for run-time selection of the right mode of control. But truly adaptive control systems using learning algorithms during their life cycle have not been utilized in commercial deployment. The reason is that the

certification process for adaptive flight control software has not yet been decided upon (Jacklin, 2008). In a different field, the regulatory regime currently used by the U.S. Food and Drug Administration (FDA) to approve medical devices is becoming too lengthy and will be prohibitively expensive with increased complexity (Lee et al., 2012).

The traditional approach of guaranteeing software reliability is fault avoidance, which uses formal specification and verification methods, as well as rigorous software development processes such as the DO-178C standard for flight control software. Verification is performed by reviews, analyses or tests. The first two steps are purely intellectual, while the latter basically includes the execution of the program to be verified and the checking whether the results of this execution are those expected (Souyris et al., 2009). Certification is estimated to consume more than 50% of the resources required to develop new, safety-critical systems in the aviation industry (Baheti, and Gill, 2011). As system complexity grows, the space of scenarios, modes, and conditions that must be tested also grows exponentially. According to the results of the recent Air Force-funded Verification and Validation of Intelligent and Adaptive Control Systems (VVIACS) Program, design-time testing already consumes 27% of the total system development costs. According to the VVIACS study, emerging flight control system development may increase V&V to 67% of total development costs without using new technologies (Schierman et al., 2008).

The paper of Graessler introduced a new V-model for engineering smart system (Graessler, 2018). The applied approaches consider the specificities of smart systems. In the engineering practice, requirements and the required parameter values change during the product development process due to external as well as internal reasons. This means that requirements cannot be fixed at the beginning of a development project. Model-based engineering approaches have been developed in research and become widely used (de facto standard) in the industrial practice. As shown in Fig. 2, modelling & analysis should cover the entire product life-cycle. Consequently, the proposed new V-Model is coupled with a product life cycle model. The extended V-model represents the inherent logical relationships between the tasks, but without specifying the sequence of the steps of the process (Graessler, 2018).

For open adaptive systems, it is typically impossible to predict their structure or behavior completely already in the design phase. Schneider and Trapp (2011) showed how to operationalize the concepts based on conditional safety certificates and corresponding run-time analyses with dynamic safety evaluations. It was also pointed that, to make the autonomous vehicle scenario a reality, human

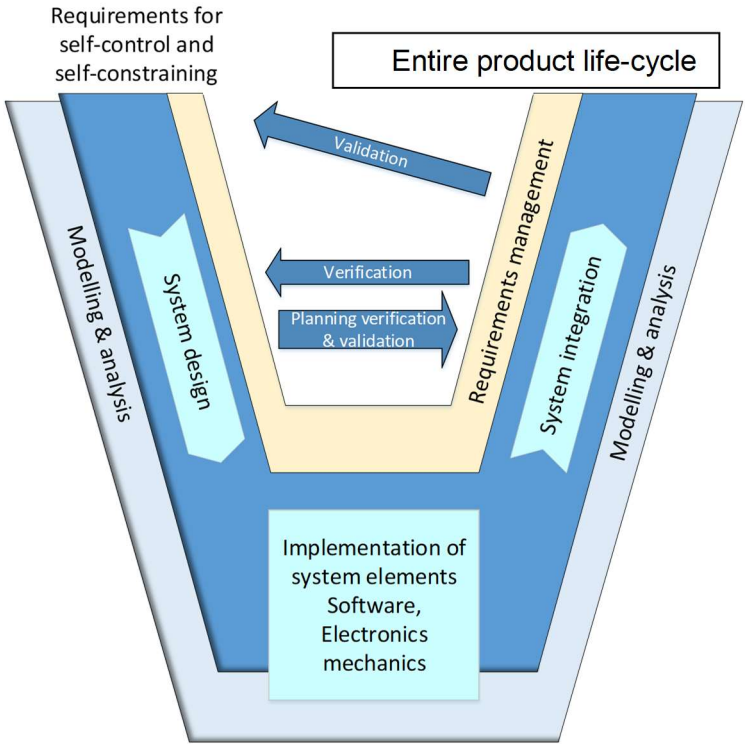

Fig. 2. V-model proposed for engineering smart systems (Graessler, 2018).

monitoring should be replaced by a certified bounding algorithm that is able to provide 'absolute' guarantees on the safety of the vehicles in a highly dynamic environment, such as urban streets (Clark et al., 2013). Critical properties and lower limits on performance can be ensured by using formally verified: (i) complexity controlling architecture patterns, and (ii) simple components for essential services. The application cannot be corrupted by faults of the unverified complex software components. It was found that a modular architecture can limit fault propagation and can support the verification of critical components.

Zhu and Sangiovanni-Vincentelli argued that, due to the complexity of the problems at hand, the use of automated co-design tools is necessary. They discussed both the needs and the challenges that are associated with developing modeling, simulation, synthesis, verification and validation tools for CPS co-design. According to them, modeling and simulation tools make it possible for designers (i) to capture the behavior of various cyber and physical components, (ii) to specify design requirements and objectives, (iii) to evaluate and validate design options based on simulations, and (iv) to identify design factors that are critical for synthesis and verification (Zhu, and Sangiovanni-Vincentelli, 2018).

The above succinct review of the current validation practice exemplifies that validation of complex systems is problematic from both financial and reliability points of view. It also points the fact that the progress is very limited with regard to validation of smart CPSs, which have the ability of adapting themselves or evolving in the exploitation phase of their life-cycle. It is emphasized by a number of authors that new validation principles are needed, which takes smartness and self-adaptability into consideration. The new solution for smart CPSs will be applicable also for complex systems with a certain level of unpredictability in environment conditions or in process parameters. Below, a short overview of the validation approaches for traditional technical systems is presented. Medical devices, which are frequently used in health care, will be used as reference cases, and as subjects of analysis and comparison.

2. Validation of traditional technical systems

A large part of medical systems manifests as assistive medical devices. Their verification and validation processes should be executed in the design phase. The aspects and measures are specified by various regulations and specification (Cooper, 2017). The processes can be generalized as shown in Fig. 3. The regulations intend to operationalize the so-called current good manufacturing practice (cGMP), which includes the Validation Master Plan as the background document of validation (Fig. 3) (FDA, cGMP, 1996). This document is company-specific and provides guidelines for engineers in terms of how to execute validation fast and in a reliable way.

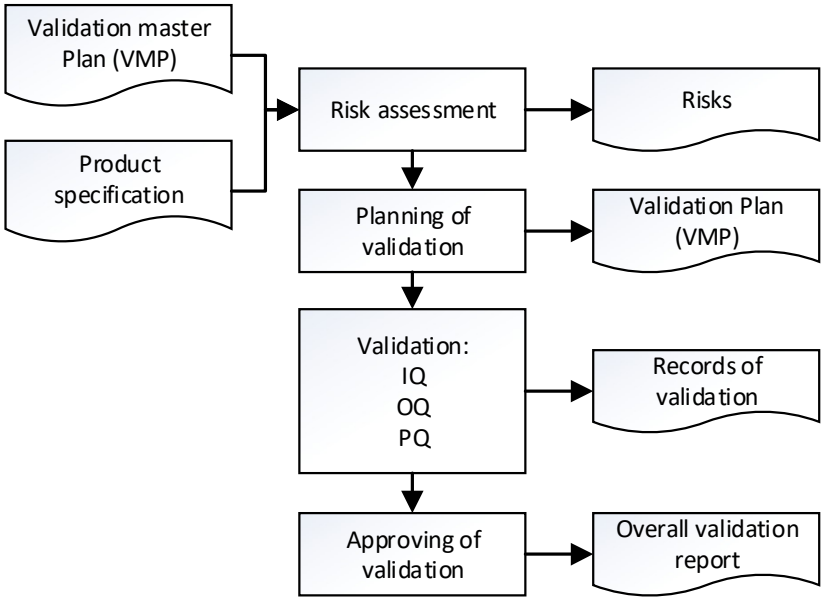

Fig. 3. Validation procedure of traditional engineering systems.

First, for each product and process, a specific validation plan needs to be prepared that contains a detailed specification of what to validate and how to validate. The most important inputs for constructing a validation plan are the specification of the product design and the assessment of risks. ISO 14971 provides a risk management frame work for manufacturers to predict the probability of occurrence of risks and their consequences (Teferra, 2017). The functions of the system, the users' requests, and the responses expected from the system need to be described unambiguously. As a result of the risk assessment, information about the recognized weak points of the system is available. Though sub-systems and components of the system may have been tested in themselves, often there is a need to retest them as active constituents embedded in the system. The validation plan should identify those operation modes of the system, which may be critical for the operation of these constituents. A proper validation plan should also determine how a system, as a whole, needs to be validated.

As a second step, the validation team has to execute the validation plan and examine if the system responds with the predefined outputs on the specific inputs. The response of the system is expected to be deterministic and to fall inside the predefined limits. As an essential part of quality assurance, equipment validation is often split into installation qualification (IQ), operational qualification (OQ), and performance qualification (PQ). IQ verifies that the qualified instrument or equipment, as well as its sub-systems, have been delivered, installed and configured in accordance with the manufacturer's specifications or installation checklist. Once each IQ protocol has been met, OQ is performed to check if the performance of the equipment is consistent with the specified user requirements, within the manufacturer-specified operating ranges. During OQ, all items included in the test plan are individually tested and their performance is documented. PQ is the final step in the equipment qualification process. This step involves the verification and documentation if the equipment works correctly and reproducibly within a specified working range. Before a final confirmation of validity, corrective actions need to be implemented concerning all potential non-conformities.

There have been various methods developed to improve the validation process. For instance, the Design for Validation (DFV) V-model promotes thinking ahead to capture all verification and validation requirements. A good validation design practice extends to checking if the requirements are verifiable (Alexander, and Clarkson, 2002). Model-based generative techniques have also been developed, which allow performing verification early in the design process, and extending the guarantees of verification to the implementation through code generation (Rajhans et al., 2009). The literature provides evidences that conventional validation approaches work well in the case of less complex and deterministic systems. Validation is conducted systematically, but based on the assumption that specific input parameters shall generate known outputs. If the system complexity increases, then difficulties may be encountered. It is difficult or expensive to check all possible inputs and working conditions. A partial solution is to focus only on those critical situations that are recognized during risk assessment. Validation cannot consider all possible combinations in the various system operation modes. If any component of the system fails, or any environmental parameter changes, then a new situation is caused that is still to be validated.

3. Approaches to validation of CPSs

In the current industrial practice, the first generation, plant-type, self-regulating CPSs are dominant. There is only a very limited number of second generation CPSs, which indeed have the ability of smart self-adaptation. Even much less has the ability of self-evolution in the exploitation phase of their life cycle. Validation of such systems is recently recognized as a challenging topic. It is understood that the road to validation approaches for the second generation CPSs leads through the validation for the first generation CPSs. In the rest of the paper, we also adopt and follow this rationale. First, we overview the procedural elements of validation for the first generation CPSs. Later on we focus on the issues of validating second generation CPSs and some of these procedural elements will be directly applied, or applied with adaptation to smart CPSs.

3.1 Run-time monitoring of operation of control sub-systems

Monitoring the states, operation condition, and performance indicators of the first generation CPSs is a starting point of validation. ModelPlex provides correctness guarantees for execution of CPS operations at run-time. It combines offline verification of CPS models with run-time validation of the

execution of system operation in compliance with the system model. If the observed behavior no longer complies with the model (that is, the offline verification results no longer apply), ModelPlex initiates provably safe fallback actions (Mitsch, and Platzer, 2014). Other promising concepts have also been proposed based on conditional safety certificates and corresponding run-time analyses (Cheng et al., 2014). Operationalization of some of these concepts is done by transforming the safety certificates into suitable run-time models, and by dynamic safety evaluations (Schneider, and Trapp, 2011).

It has been proposed that performing complete safety analyses or complex quality checks at run-time is not reasonable. It would require too much intelligence and creativity to be performed autonomously by a system (Schneider, and Trapp, 2011). Current-generation controllers are reaching a level of complexity that makes verification and validation (V&V) complicated. Nevertheless, the existing approaches are inadequate for next-generation of intelligent systems. One possible solution is to combine run-time monitoring of advanced components with simple backup modules that provide a safe reversionary mode if undesirable behavior is detected. Such an architecture allows the V&V to be partitioned into design-time V&V (for the relatively simple monitoring and fail-safe subsystems), and run-time V&V (for the fully complex controller) (Schierman et al., 2008).

Wei et al. presented a supervised reinforcement learning-based framework for controlling longitudinal vehicle dynamics by a cooperative adaptive cruise control sub-system (CACC). Incorporating a supervised network (trained by real driving data) into the actor-critical framework, improves the success rate of training. Furthermore, the driver's characteristics can be learned by the actor to achieve a human-like CACC controller (Wei et al., 2018). There are several validation challenges for adaptive control, including proving convergence over long durations, guaranteeing controller stability, using new tools to compute statistical error bounds, identifying problems in fault-tolerant software, and testing in the presence of adaptation (Jacklin et al., 2004).

Kloes et al. (2018-2) proposed an algorithm that efficiently calculates the quantitative "distance" between a system state and the system goals. The various goal types, context-dependent importance, and dependency relations are organized in a hierarchical goal model. Due to its modular structure, goals can easily be added, removed and changed at runtime. The proposed approach enables the measurement of the impact of autonomous decisions at runtime, and the efficient runtime management of changing system goals. Jiang et al. (2017) conducted a study on applying runtime verification to cooperate with current decision support system (DSS) based on real-time data. A runtime verification technique is proposed and formalized to strengthen the medical DSS. It combines formal methods in software engineering and practice guidelines in medicine to rigorous verify runtime temporal properties automatically.

Koelemeijer (2018) research aims to develop an evaluation methodology, which is capable of evaluating systems by reviewing dynamic architectures in real time. The evaluation methodology relies on a tool and an assurance case argument patterns catalogue, which enhance the automated construction and evaluation of assurance cases to determine the performance of critical adaptive systems. Pinisetty et al. (2017) introduced a method for predictive runtime verification of timed properties where the system is not entirely a black-box but something about its behavior is known a priori. A priori knowledge about the behavior of the system allows the verification monitor to foresee the satisfaction (or violation) of the monitored property.

3.2 Run-time monitoring of operation of software

Software means are the enablers of and can be a source of inspiration for CPSs, in which several software constituents need to operate without interruption. However, run-time monitoring and control are needed to achieve this (Morin et al., 2009). In the context of highly reliable and/or safety critical systems, one would actually want to monitor the execution of programs during operation and to determine whether it conforms to the specifications. Any violation of the specifications can then be used to guide the execution of the program into a safe state (Rosu, and Havelund, 2005). In the case of Monitoring-Oriented Programming (MOP), run-time monitoring is encouraged and supported as a fundamental principle for building reliable systems. Monitors are automatically synthesized from specified properties and are used in conjunction with the original system to check its dynamic behaviors.

When a specification is violated or validated at run-time, user defined actions will be triggered, which may concern any code (e.g. information logging or run-time recovery) (Meredith et. al., 2012). Models@run.time systems are a new class of reflective systems, which are characterized by tractability (due to abstraction), and their ability to predict certain aspects of their own behavior in the future. Of particular interest is the possibility to realize safe adaptive systems (Aßmann et al., 2014). The key problem of such systems is the contradiction between safety and adaptiveness. To ensure safety, all variants of the software system have to be checked against possible threats, usually at design time. In highly adaptive software systems, the number of system variants usually grows exponentially. This extends the safety check to an unfeasible degree. The reasoner component of models@run.time systems allows postponing safety checks to the run-time of the system (Aßmann et al., 2014).

The commonly used wireless links among smart software objects or towards the Internet are typically slow and subject to high packet loss. Such operational characteristics pose challenges in terms of both software running on smart objects and the network protocols, which smart objects use to communicate (Zikria et al., 2018). Mukherjee et al. have developed an end-to-end security middleware between edge devices and a core cloud of an IoT application system. The proposed middleware of the cloud-fog communication platform is based on a flexible security scheme tailored to application needs. Additionally, the intermittent security copes with unreliable network connections (Mukherjee et al., 2015).

Automotive cyber-physical systems (ACPS) should deal with joint challenges of heterogeneity, dynamics, parallelism, safety, and criticality. Xie et al. presented a fairness-based and adaptive dynamic scheduling algorithm on multiple-functional mixed-criticality. Directed acyclic graph (DAG) is used as a task graph to represent distributed automotive functions and to construct a model according to the characteristics of the concerned ACPS (Xie et al., 2017). Another reference, the AUTOSAR adaptive platform is a heterogeneous platform that integrates parallel computing, real-time requirements, safety, criticality, and existing functions into a heterogeneous architecture (Fuerst, and Bechter, 2016).

Verification of a cyber-physical system is done from a schedulability analysis perspective that is the main target in real-time theory (García-Valls et al., 2018). Kloes et al. extended the Monitor-Analyze-Plan-Execute—Knowledge (MAPE-K) feedback loop architecture by imposing a structure and requirements on the knowledge base and by introducing a meta-adaptation layer. This enables a continuous evaluation of the accuracy of previous adaptations, learn new adaptation rules based on executable run-time models, and verify the correctness of the adaptation logic in the current system context (Kloes et al., 2018-1). García-Valls et al. (2018) presented a solution for designing quasi-adaptive CPSs by using parametric models that are verified during the system execution, so that adaptation decisions are made based on the timing requirements of each particular adaptation event.

Inckia and Aria (2018) elaborated on the utilization of a model-based runtime monitoring approach for providing reliable service. Message sequence charts is used, later it allows the practitioners to express expected behavior of an application in terms of complex-event processing patterns. The presented approach is enabling a non-intrusive monitoring of IoT behavior at runtime. Pradhan et al. (2016) described the work on improving runtime support for autonomous resilience via self-reconfiguration. A runtime infrastructure governs the self-reconfiguration mechanisms. Adaptation methods starts with goal setting.

3.3 Increasing validity by fault-tolerant techniques

In the case of complex and highly automated systems, the probability of faults is bigger. A conventional feedback control design may result in an unsatisfactory performance in the event of malfunctions in the actuators, sensors, or other components (Sollazzo et al., 2010). It is necessary to design control methods that ensure nominal performance, when the occurrence of faults is taken into account (Noura et al., 2009), (Hwang et al., 2010). Fault tolerance in the flight control system of the Airbus 320/330/340 family is based on an error detection and compensation technique, using the N-self-checking software approach. Each flight control computer is designed to be self-checking, with respect to both physical and design faults, to form a fail-safe subsystem (Powell, 2011). The primary goal of the fault tolerant aircraft flight control is to recover or maintain safe flight, when failures have occurred. However, due to the significant changes that may take place as a consequence of failure, the

aerodynamic model of the aircraft will obviously be different from the nominal one. In order to enhance the capabilities of automatic flight control systems, failures will have to be detected and identified on board during the flight in real-time (Chu et al., 2010).

The Simplex architecture is an often used approach to increasing validity by a fault-tolerant technique. Its strategy is to support the use of simplicity to control complexity. The architecture is based on a simple and reliable core component that ensures the fulfilment of the critical functions of the system despite the failure of non-core software components (Sha, 2001). "Use but do not depend" is a key principle of the Simplex architecture, which minimizes the use of safety critical components, while maximizing the safe utilization of non-critical components (Sha, and Meseguer, 2008). Thus, non-safety critical devices can be added, modified, and replaced in the Simplex architecture without jeopardizing safety, provided that the architecture design rules are followed.

Lin et al. developed a resilient monitoring and control (ReMAC) system by using a previously developed resilient condition assessment and monitoring system and a Kalman filter-based diagnostic method, and integrating them with a supervisory controller. The ReMAC system is able to make correct plant and component health assessments despite sensor malfunction and make decisions (Lin et al., 2014). There exist two radically different approaches to design-fault tolerance according to the goal sought: (i) to prevent the failure of a task from causing the failure of the whole system, and (ii) to maintain service continuity (with so-called 'design diversity') (Powell, 2011). The protection scheme called Run-time Enhancement of Trusted Computing (RETC) enhances trust in CPSs containing untrusted software and hardware. RETC is complementary to design-time verification approaches. This policy-based approach is based on decoupling policies from system-specific implementations and optimizations. Interface guards enable in-line monitoring and enforcement of critical system computations at run-time. Trust is required in only a small set of simple, self-contained, and verifiable guard components (Farag, 2012).

"Golden rules" used at Airbus to design a fault-tolerant aircraft combine a stringent development process, rules for hardware design, run-time monitoring, and a safe-mode of operation (Goupil, 2010). Abid et al. (2014) discussed the application of the TS fuzzy approach to fault detection in CPSs. The current methods of performing software and system verification and validation require exhaustive offline testing of every possible state space scenario. However, this is an impossible task for adaptive, non-deterministic, and near infinite state algorithms. The approach proposed by Alho and Mattila (2015) utilises the loosely coupled nature of services to implement fault tolerance for CPSs. Based on the concepts of fault isolation and recovery at the service level, the approach is without significant overhead.

According to Gerostathopoulos et al. (2019) failures may appear when self-adaptive a CPS operates under environment condition, which they are not specifically designed for. The homeostasis of CPS was improved by introducing run-time changes to the architecture-based self-adaptation strategies according to environment stimuli. Four mechanisms that reify the idea was introduced: (i) collaborative sensing, (ii) faulty component isolation from adaptation, (iii) enhancing mode switching, and (iv) adjusting guards in mode switching.

3.4 Platforms, smart components, and decentralized control

Several efforts are documented in the literature that strived after the development of enablers and facilitators of system validity. Below we concentrate on issues concerning frameworks, platforms, smart components, and decentralized control. Zheng, Deng et al. (2016) presented a cross-layer codesign framework that addressed the tradeoff between security and control performance, while guaranteeing platform schedulability for CPSs. Security techniques, such as message encryption, are applied to mitigate risk. The CONVINCE framework, proposed by Zheng, Lin et al. (2016), included mathematical models, synthesis, verification and validation algorithms, and a heterogeneous simulator in a holistic environment. It considered a variety of design options with respect to constraints and objectives across system layers, and took into consideration of environment disturbance and possible security attacks.

The genuine idea is to divide the problem of verifying CPSs into two collaborative sub-problems: (i) functional verification under constraints and invariants guaranteed by synthesis (e.g., verifying safety property when assuming sensor-to-actuator delay is within certain bound, and (ii) software/hardware

platform synthesis under constraints and invariants specified by verification on sensor-to-actuator-delay. Seiger et al. (2015) presented an object-oriented workflow language for formalizing processes within heterogeneous and dynamic environments. The proposed component-based meta-model enables a hierarchical composition of processes as well as of process variants.

Even conventional CPSs often include smart components that use local adaptation to improve the performance of the whole system or to provide response in case of damage. Gallagher et al. (2017) argued that evolvable and adaptive hardware (EAH) components are enabling means for smart system development. They presented an approach that addresses many verification and validation questions related to the use of EAH smart components. Our observation is that the cyberware constituents (knowledge, ontologies, and reasoning mechanisms) have not received attention yet from the perspective of ensuring system validity and faultless operation.

It has been experienced that centralized control is hardly adequate to manage the whole of complex, multi-actor, distributed systems. On the other hand, self-adaptation was recognized as an effective approach to manage the complexity and dynamics inherent to CPSs. In the case of large and heterogeneous systems, multiple MAPE components might coordinate with each another to decentralize the control (Weyns et al., 2013). D'Angelo et al. (2017) proposed MAPE-K components as first-class modeling abstractions and provided a framework supporting the design, development, and validation of decentralized self-adaptive cyber-physical systems.

Proposed by Zavala et al. (2018), the Smart Adaptation through Contextual REquirements (SACRE) approach is the combination of a successful technique for supporting the adaptation of self-adaptive systems contextual requirements in the presence of runtime uncertainty and extended well known architectural styles for enabling the application of the adaptation technique in self-adaptive systems. Zheng et al. (2017) argued that existing formal method techniques and simulation are insufficient for supporting the development of entire general-purpose CPS. The current state of the practice in CPS verification and validation remains an ad hoc trial and error process. There are still significant gaps between the formal models of computing and the formal models of physics that underpin today's CPS systems.

Based on the above concise survey, we can conclude that run-time monitoring, reliable operation principles, and fault tolerant technics are already known enablers in the context of conventional CPSs. However, transferring these to the domain of smart and self-adapting systems is not straightforward. The mentioned systems often operate in unpredictable environments and therefore in a not deterministic manner. Faults of critical or less critical components, disturbances in network connections, deviations from the set operational targets are the realistic operation conditions for them. Emergent faults and variation of the operation parameters can hardly be anticipated and encountered during the development phase of these systems. Therefore, only a limited subset of the existing validation approaches and methods are directly applicable to smart CPSs. It needs further inquiries and investigations to learn which approaches can be in one way or other adapted to the needs of validation of smart CPSs.

These studies are necessary for the reason that the level of unpredictability and adaptation capability is much higher in the case of these systems. on the other hand, transdisciplinary knowledge may also be helpful. For instance, it is expected that, having fewer restrictions, the results of the research in validation of unmanned vehicles for military purposes will make a significant contribution to the validation of unmanned vehicles for civil applications (Finn, and Scheding, 2010). It is also an issue how the self-awareness building, self-learning and self-evolving abilities can be utilized for operation validation in the exploitation phase and how dynamic behavioral models can be autonomously generated to maintain purposeful operation under various circumstances. It can also be concluded that the issue of run-time resource management and using this affordance in providing adaptation solutions is not addressed in the current literature.

4. Toward validation of a smart CPS

4.1 The procedural framework

Smart CPSs implement a higher level of integration of hardware, software, and cyberware

technologies than any other type of systems previously (Horváth, and Gerritsen, 2012). The systems knowledge also plays a crucial role in that. Smart CPSs are typically deeply embedded and having numerous internal/external relationships with natural or engineered environments. Though smart CPSs are open systems from both architectural and operational viewpoints, they implement a high level of synergy with regard to all dynamically changing functionalities and components. Based on the collected data, they can build awareness of the states and contexts of operation. In addition, on the basis of this acquired awareness, they can better interact with humans cognitively/semantically and influence their personal routine and/or social life, which in turn may influence the supervisory control provided by them (Marwedel, 2011). These all underline our motivational hypothesis that smart CPSs are not deterministic systems, therefore their operation cannot be validated in the conventional ways. This gave floor to the research question: What is a dedicated validation approach (a validation methodology or a set of methods) for performance validation of smart CPSs?

It cannot be expected that fully-fledged and detailed validation procedures dedicated to smart CPSs will be implemented overnight. Only a step-by-step evolution comprehension and advancement can be anticipated. In order to contribute to the discussions about this important issue and to the progress in this context, below we propose a framework procedural for a systematic performance validation of smart CPSs. It is shown in Fig. 4. This procedural framework includes seven steps, which are all analytic and prescriptive in nature. The conventional validation approach tries to forecast and regulate the system's operation based on the knowledge and experiences available concerning the developed system. However, the proposed approach intends to share the tasks of validation between the designers and the concerned systems. It means the overall validation process includes a design phase validation (DPV) part, which is executed by experienced designers/engineers, and a run-time phase validation (RPV) part, which is done during operation by the system. The resources needed for this purpose may be availed by the developers in the design phase, together with the resources needed for adaptation of the system, and/or may be obtained by the system during its operation. In addition, procedurally, it suggests a set of activities for 'verification of properness' (VoP), and a set of activities for 'validation of appropriateness (VoA).

The outcome of the literature review presented in the previous section was used as a starting point for conceptualization of the procedural framework shown in Fig. 4. Furthermore, the steps of validation

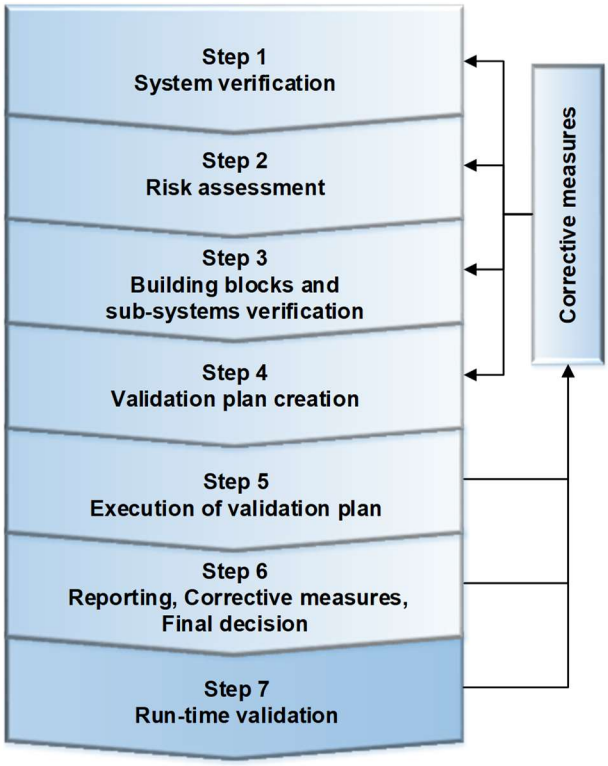

Fig. 4. The procedural framework of performance validation of smart CPSs

that are used in the case of complex systems and conventional CPSs were considered as the basis of the concept of the validation process. Specific attention was paid to checking the smartness and reliability of the building blocks and the system as a whole. Based on the explanation provided above, it is evidential now that run-time validation-specific additional functionalities and context dependent reasoning mechanisms should be included in smart CPSs, such as objective-sensitive self-monitoring mechanisms, self-constraining and self-supporting mechanisms, and other enablers, e.g. run-time validity evaluator and resource manager.

Additional information concerning the identified steps of performance validation process will be provided in the next sub-sections. However, it could not be intended to provide all low level information about operationalization and implementation of the procedural framework. For the sake of conformity, we use the terminology and definitions according to FDA, cGMP, 1996. System verification is understood as confirmation by examination and provisioning objective evidence that the specified requirements have been fulfilled (i.e. the system is what it is supposed to be). System validation means establishing objective evidence that the system specification conforms to the user needs as well as to the intended use (i.e. the system performs what it is supposed to do). Verification precedes validation to ensure that the user needs and the intended usage can be fulfilled in a consistent manner.

4.2 System verification

A prerequisite of a successful performance of a system is a correct theoretical underpinning and a proper system design that provides reliable and fault-tolerant operation. The goal of verification is to check and safeguard these characteristics. In an optimal situation, this extends also to the investigation of the design principles and the potential risks. During verification of the system design it is checked if (i) the requirements were comprehensively defined and considered in the design process, and (ii) proper design practice and design principles were applied. This can be done by complying with a detailed check list. There are several examples for constructing and using detailed check-list in the daily practice. A relevant example is the check-list used at Airbus, which includes rules for designing a fault-tolerant aircraft (Goupil, 2010).

The considered basic design principles are such as hardware redundancy, real-time monitoring, safe-mode operation, dissimilarity, and robustness with regard to different disturbances. The check-list in Table 1 is an extended version of the check-list mentioned above. The extension captures the specific expectations towards smart CPSs, which were identified in the literature review and included in Table 1 rows 8, 9 and 10. The check-list is supposed to give answer to the question how well the development

Table 1. Check-list for verification of smart CPSs

No.	Verification of the principles of system design
1	System safety assessment and investigation of the effects of each functional failure
2	Formal development process with planning, development, system modelling and simulation, verification, configuration management, quality assurance issues
3	Hardware redundancy for all safety critical building blocks
4	Real-time monitoring of sensors, computers, and system operations
5	Switching into safe mode operation in real-time in the case of failure or any safety critical building block or sub-system, or in the case the system operates outside predefined limits; application of simplex architecture
6	Application of dissimilarity of a critical subsystem (two different computers and software) and installation segregation
7	Robustness of system and equipment on environmental disturbances
8	Implementation of self-constraining rules in the case of self-evolution
9	Run-time validation function (self-checking of safety critical situations)
10	Tools and methods for self-awareness; the capability to adapt to new circumstances

process was informed and conducted. A systematic development process that apply appropriate tools and methods in multidisciplinary team get to better results.

System verification is actually not part of validation, but it is a necessary prerequisite. A system without robust architecture and without required functionality cannot pass validation test. Each smart CPS has to have capability of run-time validation that needs to be executed regularly during the exploitation phase of the system. Smart CPSs can in general evolve during the life-cycle therefore they have to be able to constrain themselves. In the case of changes in environment the system needs self-awareness functions.

4.3 Risk assessment

The proposed second step of the performance validation process is a combined risk analysis (exploration) and assessment (evaluation) (Fig. 4). Risk assessment is an important part of each validation process. Obviously, all possible cases cannot be checked in real life, even if automatic execution of tests and control are possible. On the other hand, it is required to recognize all safety critical situations and assess their consequences. By a proper system configuration, and design and real time monitoring of the consequences of such situations have to be minimized or even eliminated. In addition, the risk assessment should take into account (i) the fault of critical components, (ii) the faults of the infrastructure and communication, (iii) misuse by end-users, (iv) changes in environment, and (v) the changes due to system adaptation and evolution.

The communication channels and possible cyber-attacks need special attention. The outcome of risk assessment forms a kind of background (context) in the next steps of the validation plan. In the case of a smart CPS, self-control, adaptation, robustness of operation and safety also need special attention. The main issue is whether the system mechanisms for self-constraining are appropriate and whether they will function properly even in the case of adaptation and/or reconfiguration of the CPS? Additional research is needed in this area. In the near future it is realistic to count on a safety-critical super control mechanism, which allows system adaptation only within predefined limits (and do not violate the operation of already verified safety mechanisms).

4.4 Verification of the sub-systems and building blocks

CPSs and smart CPSs are complex arrangements that consist of a large number of sub-systems, building blocks, and components and that may lend themselves to different configuration options and application combinations. The entirety of a complex system cannot be checked for all arrangements and settings. The current trend is to manage and control the complexity of the system by applying smart and autonomous building blocks. The system as a whole has to detect the state of all sub-systems and components in real time. It means that some form of supervisory smartness is needed on both sub-system and component levels. The sub-systems and the components need to communicate with each other. Interoperating sensors can inform the system about the state of operation, and provide data for setting the values of the system/component parameters.

In the overwhelming majority of cases, a number of key building blocks determine the functionality and safety of the whole system. This entails that verification of the sub-systems and building blocks may be reduced to confirmation of the reliability of these critical components. The critical components have to be recognized and additionally verified. However, the consequences of possible faults need to be checked. Classification of building blocks based on their operational criticality enables focusing on those that need more attention from the perspective of a safe operation. Associated with this is the question about the level of intelligence and autonomy of building blocks. Paradoxically, it reduces the complexity of validation if verification can be done on the component level, but it does not guarantee higher confidence in the reliability of the entire system. For instance, in terms of safety, only certain level of probability can be expected.

4.5 Constructing a validation plan

A validation plan is based on system specifications, risk assessment, and criticality classification of the building blocks. The plan explains how to check the key sub-systems and components that enable the overall functioning of the system and provides safe operation. It is not possible to cover all

combinations of configurations and working conditions by the validation plan. It can focus on the mechanisms that enable self-control and resilience against faults. Table 2 presents the list of activities that need to be included in a validation plan.

4.6 Execution of validation

The validation team needs to have competences for an objective and critical assessment of a developed system, and they need to be independent from the development team, at least concerning the final approval. The process of validation needs to be organized according to the approved validation plan. The completion and the results of each validation step have to be well documented. Manual step-by-step checking of smart CPSs is not possible in most cases because of complexity. Validation procedures need to be executed as a set of commands written in a script file. The validation team is, first of all checking the system responses and approve generated records. The experts need to assess if self-awareness and system resilience to environment disturbances works well. The system shall behave according to specification in safety critical situations. The core of the self-validation procedure is conducted autonomously; it has to be part of the system.

The requested functionality of smart CPSs is run-time validation. The system has to demonstrate that it is able to adapt to new circumstances and new configuration in run-time. Validation procedure needs to contain elements of random generation of testing conditions to simulate realistic circumstances for real application.

4.7 Reporting, corrective actions, and a final decision

New findings shall trigger additional checking or corrective actions. Validation can be repeated in several iterations. At the end, the validation team needs to approve that the smart CPS has capabilities of self-control and self-validation even in new, at this moment unknown circumstances. It is a challenge to approve such capabilities. Real technical systems always have some residual risks. The validation team has to take a decision if advanced solutions have been used to maximize safety, and if design validation and risk assessment considers different aspects of application. In adaptive and non-deterministic systems, we cannot talk about absolute safety. The new paradigm requires a change of mind from deterministic and hierarchical controlled systems into adaptive and autonomously controlled mechanisms. The new approach has to be considered at taking decisions for system validation.

4.8 Run-time validation

Formal validation at the end of system design is only the first validation step in the system life cycle.

Table 2. Check-list included in a validation plan for smart CPSs

No.	Activity
1	Installation qualification; verification of building blocks – are they according to specification; Is capacity of the infrastructure appropriate?
2	Checking of the system functions under normal conditions; does it work according to specifications? Checking system performance at overload
3	Checking of run-time monitoring functions; short and long term
4	Does the save-mode of operation work? Checking of switching to the safe-mode from recognized critical situations in real-time
5	Checking of system resilience in the case of failure, or any safety critical building block or sub-system
6	Checking of system resilience in the case it operates outside of predefined limits, and in the case of environment disturbances
7	Checking of self-constraining mechanisms. How does the system control its own evolution?
8	Checking of safety critical situations. If the system is resilient to the worst cases, it shall work well under normal conditions. It is important to define such critical cases that cover all others situations.
9	Checking Tools and methods for self-awareness and checking the capability to adapt self-control to new circumstances. This is a part of the self-validation.
10	Checking of the self-validation module in real time (run-time validation) or off-line; according to the specification
11	The operator or end-user can give the system wrong instructions; how safe and resilient is the system in the case of such failures?

Table 3. Comparison of validation principles between traditional medical devices, first generation CPSs, and smart CPSs

	Validation principles		
Criteria	**Traditional medical devices**	**First generation CPSs**	**Smart CPSs**
Component level	Verified components	Smart and self-aware components	Smart and self-aware components, smart sub-
System design	Numerical simulations, functional testing	Design for validation, System simulation	Algorithms for self-control constraints
Verification and validation methods	Design verification, Risk assessment, Validation plan, Life span testing	Design verification, Risk assessment, Validation plan, Smart self-control	Design verification, Risk assessment, Validation criteria, Smart self-control, Self-constraining
Execution of validation	Execution of validation plan	Fault tolerant testing, Switching to safe mode of operation, Run-time monitoring, Run-time validation	Fault tolerant testing, Switching to safe mode of operation, Run-time monitoring, Run-time validation, Testing of self-constraining algorithms
Validation during system exploitation	Components checking, Periodic functional tests, Periodic maintenance	Run-time monitoring, Functional tests of sub-systems, Switching to safe mode operation	Run-time validation, Testing of self-constraining algorithms, Checking of safety criteria and development policy

The system has to prove that it is capable of self-constraining and self-validation during the whole life cycle. The key function of the smart CPS is run-time validation. A system needs to be able to execute run-time validation in real-time and before/after implementation of any modification in system configuration that could influence the way of operation. Validation of run-time validation capability is an open topic that requires additional research. Validation needs to confirm that the system is able to adapt and self-control even outside of the current working limits. For less demanding systems self-validation can be done periodically during idle periods. In the advanced systems that require continuous operation, run-time validation has to be executed in real-time while the system works as part of regular system monitoring and checking.

Figure 5 presents a generic model for run-time validation. It is based on run-time monitoring,

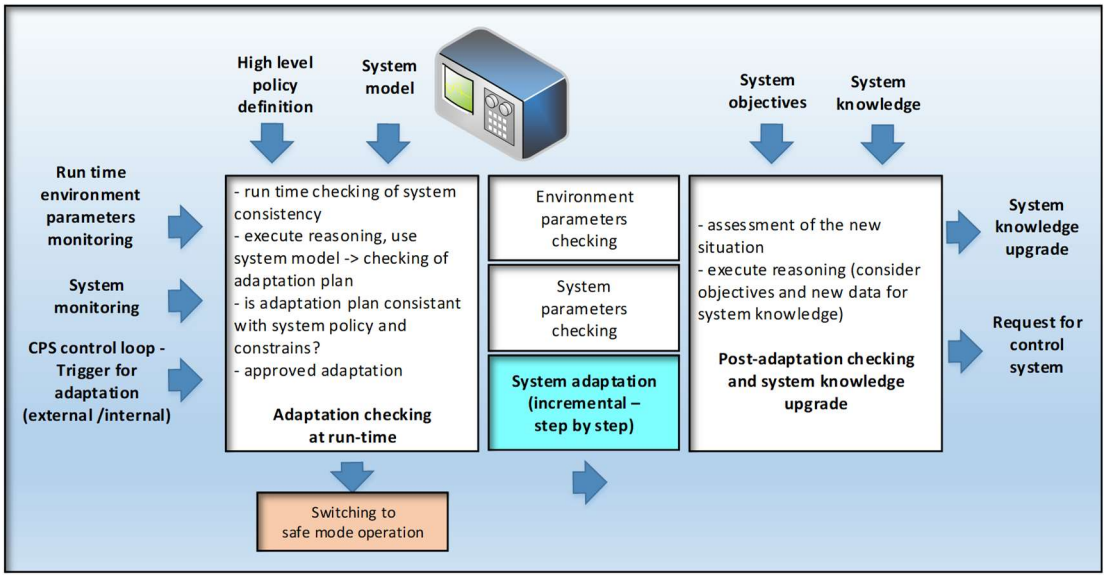

Fig. 5. General model for run-time validation

reasoning about system behavior and incremental execution of adaptation. The CPS need to be resilient to changes in environment and have capability for self-evolving. The feedback loop that can react on changes in environment in real-time is the core of the system. The monitoring system controls input parameters and system situation in real time. Some of input parameters can be set in deterministic way, others are random and depend on the context. The activities presented in Fig. 5 are executed concurrently and in small incremental steps that enables system control in each moment.

Adaptation can be done (i) in the first control loop or (ii) in the second self-enhancement loop. In the case that environment parameters trigger control loop, adaptation plan is prepared first. Control parameters are checked according to system policy and system consistency. In the case the adaptation is approved new setting of parameters is conducted. The preferred mode of parameters setting in incremental change from current value to the target one. System and environment parameter values are monitored during such incremental adaptation. In the case of inconsistency, the system can be switched to the save mode of operation or it can be rollback to previous state. Continuous monitoring and prompt reaction can avoid safety critical situation even it is a new combination of parameters that have not been verified before.

Each control parameter adaptation is included into the learning loop. The system situation is assessed after each parameters setting and system knowledge is after that upgraded with a new combination. During the exploitation phase of the CPS the system knowledge is expanding in small steps. In the case a system and safety policy allows extension of the parameter range, a new knowledge enables system operation under new circumstances that have been not validated in the design phase.

The proposed run-time validation model is based on context and system awareness and on self-learning capability. The system needs reliable detection of critical situations in real-time and switching to the safe mode of operation. That mean safety need to be integrated into the system from the beginning.

5. Discussion

CPSs can be split into first generation, second generation, and higher generation CPSs. For transition from the first to second generation, quasi-smart CPSs are allowed to adapt only within pre-defined limits. In most cases the validation approach for conventional CPSs can be applied to quasi-smart CPSs. The adaptation details of the quasi-smart CPSs are not known in the design phase. However, they shall not threaten safety and the expected functions of the system. Adaptations of the quasi-smart CPSs in the system life cycle are some kind of extrapolation from the design phase defined basic system structure. Several design principles that make systems more reliable and fault-tolerant known from first generation of CPSs can be applied to quasi-smart CPSs. Literature and regulation review has shown very narrow evidence on validation procedures of smart CPSs. The new V-model for Engineering Smart System (Graessler, 2018) is a step in the right direction. The V-model conducts requirements setting, modelling & analysis through the entire product life-cycle. However, there are no specific instructions for validation.

The proposed model for validation of smart CPS pays attention to robust system configuration; the first step is system verification. An important message of this paper is that system designers need to be aware about validation requests through all phases of the design process. A prerequisite for validation of smart CPS is run-time monitoring and fault tolerant design. The second part is new and is based on smartness and self-constraining. In the section 4.8, the model proposed for run-time validation is presented. What are open issues of implementation? Run-time validation is based on real-time reaction in the case of new circumstances. That means safety depends on timely execution of all operations. System configuration shall guaranty reliable communication and prompt response in all situations and detection of critical situations. García-Valls et al. (2018) presented a solution for designing adaptive cyber-physical systems by using parametric models that are verified during the system execution, so that adaptation decisions are made based on the timing requirements of each particular adaptation event. Their approach allows the system to undergo timely adaptations that exploit the potential parallelism of the software and its execution over multicore processors.

During verification phase all critical elements need to be recognized and risk assessment shall foresee

failure of critical elements. Later system monitoring shall assure availability of all safety critical elements. It includes also elements for switching into safe mode operation. Reliability and risk analyses in the case of particular elements failure is an open issue also in the systems without run-time validation. In the autonomous CPS environment and system awareness and smart reaction is expected to avoid any uncontrolled situation.

The proposed incremental parameter setting works well in linear systems. In the case of non-linear system and discrete parameter setting additional safety mechanisms need to be implemented such as strict considering of the predefined parameters ranges. The important safety mechanism is switching into safe mode of operation. However, there are non-linear systems and situations that after passing specific limit cannot return to safe mode of operation. Risk assessment shall recognize and avoid such situations.

Designers of smart CPSs need prognostic approaches, while systems need run-time constructed validation plan generation. Truly smart CPSs evolve in their life cycle more radically, and they can also activate new resources and system functions. Such changes cannot be managed inside fixed constraints defined in the system design phase. Smart CPSs need to have the ability to set constraints to themselves, and also to evolve validation methods. The validation process needs to focus attention on this self-controlling ability. The initial settings are not deterministic control mechanisms, but more general rules: what is the requested safety level, guidelines for system evolution, and the policy for self-control. It does not mean that those general rules cannot be updated during the system life cycle. Validation process of smart CPS is complex and demanding activity. We propose future research towards a run-time validation engine that will make validation faster, more accurate and more reliable for the whole system.

This paper presents in section 4 comparison between traditional medical devices, first generation CPSs, and smart CPSs. The basic principles are the same for all kinds of systems. However, smart CPSs need specific approaches that need to be developed and tested in the future. The proposed model recognized new functions for smart systems, such as self-constraining mechanisms, run-time validation and testing of self-constraining algorithms, safety criteria and development policy. A requested feature for smart systems is the capability of run-time self-validation. That mean validation is not once only conducted procedure at the end of design process, it is a core function of the system that needs to be permanently maintained and executed.

In most cases the initial safety criteria and validation methods can be applied, that already have been defined during the design process. There is evidence in the analyzed examples that smartness has to replace complexity first in the first generation CPSs. It is not realistic to check all options and combinations in a complex CPS, and to consider all circumstances because of disturbances in environment (Filho, 2017). The context aware system with a certain level of intelligence can adapt to new circumstances much better than a deterministic hierarchical system. Therefore, we propose developing and applying of smart control first to the first generation CPSs. After proving the efficiency of smart control in several applications in practice, it will also show the way and give a certain level of self-confidence for smart CPSs.

6. Conclusions

Smart CPSs are purpose-driven and context-dependent behavior and reasoning that make architectural and functional adaptation and evolution possible. Smart CPSs cannot be validated with the conventional deterministic approaches. The traditional control that is usually based on the human in the loop principle needs to be replaced by some sort of automatism in these applications. Validation of smart CPSs is a necessary step before more spread use of smart CPSs can happen. The survey has shown a gap in advanced validation methods that could be applied for smart CPSs. The new methods and approaches have been recognized that enable validation of higher level of adaptability, system and environment awareness, and self-control of constraints and resources in real-time, and run-time validation. The contribution of this paper is comprehensive overview of the state of the art, the working proposal of validation steps for smart CPSs, and model for run-time validation.

The presented proposal for validation of smart CPSs is still far away from application from the point

of view of current practice, and legislation. Development of complex and smart CPSs is demanding new approaches. If a system has freedom for self-adaptation, then it should also be equipped with meta-knowledge and a supervisory controller that context dependent semantic reasoning about the operational and behavioral objectives and about the fulfilment of requirements possible. Smartness needs to replace deterministic control and principles of self-validation has to be further developed from theoretical and practical points of view. These radical changes need to be integrated into the validation process.

This paper is focused on the approach "done by human and done in design phase". Due to unfinished ongoing research and space limitations open issues related to "done by system and done in run-time phase" are only partly covered. The paper proposes future research towards a run-time validation engine that will make validation faster, more accurate and at the whole system more reliable. It is expected that the described transformation will happen as an evolution in several steps, and not as an overnight change. The proposed concept toward validation of smart CPSs is presenting the framework. The principals for self-constraining need to be developed and later approved in practice.

Acknowledgement

This work was supported by Slovenian Research Agency – ARRS, grant number contract no. P2-0265.

References

Abid M., Khan A.Q., Rehan M., and Haroon-ur-Rasheed (2014). TS Fuzzy Approach for Fault Detection in Nonlinear Cyber Physical Systems, Computational Intelligence for Decision Support in Cyber-Physical Systems, Vol. 540, 421–447.

Alexander K., and Clarkson P.J. (2002). A Validation Model for the Medical Devices Industry, Journal of Engineering Design, Vol. 13, No. 3, 197-204.

Alho P., and Mattila J. (2015). Service-oriented Approach to fault tolerance in CPSs, The Journal of Systems and Software, Vol. 105, 1–17.

Aßmann U. et al. (2014). A Reference Architecture and Roadmap for Models@run.time Systems, Bencomo N. et al. (Eds.), Models@run.time, Foundations, Applications, and Roadmaps, Volume 8378, Springer, 1–18.

Baheti R. and Gill H. (2011). Cyber-physical Systems, The Impact of Control Technology, T. Samad and A.M. Annaswamy (eds.), Available at www.ieeecss.org. 161–165.

Cheng B. H. C et al. (2014). Using Models at Run-time to Address Assurance for Self-Adaptive Systems, Models@run.time, Foundations, Applications, and Roadmaps, Volume 8378, Springer. 101–136.

Clark M. et al. (2013). A Study on Run Time Assurance for Complex Cyber Physical Systems, Air Force Research Laboratory, Vanderbilt University, Iowa State University, University of Pennsylvania, Galois Inc. 1–59.

Cooper B. (2017). The GMP Handbook, A Guide to Quality and Compliance, Kindle. 1–150.

D'Angelo M., Caporuscio M. and Napolitano A. (2017). Model-driven Engineering of Decentralized Control in Cyber-Physical Systems, 2017 IEEE 2nd Inter. worhshops on foundations and applications of self* systems, Tucson, AZ, USA, 7–12.

DO-178C / ED-12C (2012). Software Considerations in Airborne Systems and Equipment Certification, Radio Technical Commission for Aeronautics.

Farag M. M. (2012). Architectural Enhancements to Increase Trust in Cyber-Physical Systems Containing Untrusted Software and Hardware, Dissertation at Faculty of the Virginia Polytechnic Institute and State University.

Filieri A., Tamburrelli G., and Ghezzi C. (2016). Supporting self-adaptation via quantitative verification and sensitivity analysis at run time. *IEEE Transactions on Software Engineering*, Vol. 42, No. 1, 75-99.

Filho M.F., Liao Y., Loures E.R., and Canciglieri O. (2017). Self-aware smart products: systematic literature review, conceptual design and prototype implementation, Procedia Manufacturing, Vol. 11, 1471–1480.

Finn A. and Scheding S. (2012). Development and Challenges for Autonomous Unmanned Vehicles, A Compendium, Springer. 1–207.

Fuerst S. and Bechter M. (2016) AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR adaptive platform, 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W), 2016, 215–217.

Gallagher J.C., Matson E.T. and Goppert J. (2017). A Provisional Approach to Maintaining Verification and Validation Capability in Self-Adapting Robots, 2017 First IEEE Intern. conference on robotic computing (IRC), Taichung, Taiwan, 382-388.

García-Valls M., Perez-Palacinb D. and Mirandolac R. (2018) Pragmatic Cyber Physical Systems Design based on Parametric Models, The Journal of Systems & Software, Vol. 144, Oct. 2018, 559–572.

Gerostathopoulos I., Skoda D., Plasil F., Bures T., and Knauss A. (2019). Tuning self-adaptation in cyber-physical systems through architectural homeostasis. Journal of Systems and Software, Vol. 148, 37-55.

Goupil P. (2010). Industrial Practices in Fault Tolerant Control, Edwards C., Lombaerts T., Smaili H. (eds.). (2010), Fault Tolerant Flight Control: A Benchmark Challenge, Springer, 157–166.

Graessler I. (2018). Competitive Engineering in the Age Of Industry 4.0 and Beyond, in Proceedings of the TMCE 2018, Las Palmas de Gren Canaria, Spain, Delft University of Technology, Delft, 1–10.

Horváth I. and Gerritsen B.H. (2012). Cyber-Physical System: Concepts, Technologies and Manifestation, in Proceedings of the TMCE 2012, Karlsruhe, Germany, Delft University of Technology, Delft, 1–16.

Horváth I., Rusák Z. and Li Y. (2017). Order Beyond Chaos: Introducing the Notion of Generation to Characterize the Continuously Evolving Implementations of Cyber-Physical Systems, ASME IDETC/CIE, Cleveland, Vol. 1. 1–14.

Hwang I., Kim S., Kim Y. and Seah C.E. (2010). A Survey of Fault Detection, Isolation, and Reconfiguration Methods, IEEE transactions on control systems technology, Vol. 18, No. 3. 636–653.

Incki K., and Ari I. (2018). Model-Based Runtime Monitoring of Smart City Systems. Procedia computer science, Vol. 134, 75-82.

Jacklin A. A. (2008). Closing the Certification Gaps in Adaptive Flight Control Software, AIAA Guidance, Navigation and Control Conference and Exhibit, Honolulu, Hawaii. 1–14.

Jacklin A. A. et al. (2004). Verification, Validation, and Certification Challenges for Adaptive Flight-Critical Control System Software, AIAA Guidance, Navigation, and Control Conference and Exhibit, Providence, Rhode Island. 1–10.

Jiang Y., Song H., Wang R., Gu M., Sun J., and Sha L. (2017). Data-centered runtime verification of wireless medical cyber-physical system. IEEE Transactions on Industrial Informatics, Vol. 13, No. 4, 1900-1909.

Katz P. and Campbell C. (2012). FDA 2011 Process Validation Guidance: Process Validation Revisited, Journal of GXP Complience, Vol. 16, No. 4, 18–29.

Kloes V., Goethel T. and Glesner S. (2018-1). Comprehensible and Dependable Self-Learning Self-Adaptive Systems, Journal of Systems Architecture, Vol. 85–86, May 2018, 28–42.

Kloes, V., Goethel, T., & Glesner, S. (2018-2). Runtime management and quantitative evaluation of changing system goals in complex autonomous systems. Journal of Systems and Software, Vol. 144, 314-327.

Koelemeijer D. (2018). Enhancing the cyber resilience of critical infrastructures through an evaluation methodology based on assurance cases. Procedia Computer Science, Vol. 126, 1779-1791.

Lee I. et al. (2012). Challenges and Research Directions in Medical Cyber–Physical Systems, Proc. IEEE, Vol. 100, No. 1, 75–90.

Lin W.C, Villez K.R.E. and Garcia H.E. (2014). Experimental Validation of a Resilient Monitoring and Control System, Journal of Process Control, Vol. 24, No. 5, May 2014, 621–639.

Marwedel P. (2011). Embedded System Design: Embedded Systems Foundations of Cyber-Physical Systems, Springer. 1–281.

FDA, Medical Devices; Current Good Manufacturing Practice (cGMP) (1996). Food and Drug Administration, 21 CFR Parts 808, 812, and 820, Vol. 61, No. 195, pp. 52601–52662.

Meredith, P.O., Jin, D., Griffith, D., Chen, F. and Rosu, G. (2012). An overview of the MOP run-time verification framework. Int J Softw Tools Technol Transfer, Vol. 14, No. 3, 249–289.

Mitsch S. and Platzer A. (2016). ModelPlex: Verified Run-time Validation of Verified Cyber-Physical System Models, Formal Methods in System Design Springer, Vol. 49, No. 1-2, 33–74.

Morin, B., Barais, O., Jezequel, J.-M., Fleurey, F. and Solberg, A. (2009). Models at Run-time to Support Dynamic Adaptation. Computer. Vol. 42, No. 10, 44–51.

Mukherjee B., Wang S., Lu W., Neupane R.L., Dunn D., Ren Y., Su Q. and Calyam P. (2018). Flexible IoT Security Middleware for End-To-End Cloud-Fog Communication, Future Generation Computer Systems, Vol. 87, October 2018, 688–703.

Neumeyer S., Exner K., Kind S., Hayka H. and Stark R. (2017). Virtual Prototyping and Validation of Cpps within a New Software Framework; Computation, Vol. 5, No. 1, 2–14.

Noura H., Theilliol D., Ponsart J.-C. and Chamseddine A. (2009). Actuator and Sensor Fault-tolerant design, Fault-tolerant Control Systems: Design and Practical Applications, Springer, 7–37.

Ping Chu et al. (2010). Real-Time Identification of Aircraft Physical model for Fault Tolerant Flight Control, Edwards C., Lombaerts T., Smaili H. (eds.). Fault Tolerant Flight Control: A Benchmark Challenge, Springer, 129–153.

Pinisetty S., Jéron T., Tripakis S., Falcone Y., Marchand H., and Preoteasa, V. (2017). Predictive runtime verification of timed properties. Journal of Systems and Software, Vol. 132, 353-365.

Pradhan S., Dubey A., Levendovszky T., Kumar P.S., Emfinger W.A., Balasubramanian D., Otte W., and Karsai G. (2016). Achieving resilience in distributed software systems via self-reconfiguration, The Journal of Systems and Software, Vol. 122, 344-363.

Powell D., Arlat J., Deswarte Y. and Kanoun K. (2011). Tolerance of Design Faults. C.B. Jones, J.L. Lloyd. (eds). Dependable and Historic Computing, Springer, Lecture Notes in Computer Science 6875, 428–452.

Rajhans A., Cheng S.W., Schmerl B., Krogh B.H., Aghi C. and Bhave A. (2009). An Architectural Approach to the Design and Analysis of Cyber-Physical Systems, In the Proceedings of the Third International Workshop on Multi-Paradigm Modeling, Denver, CO. 1–10.

Rosu G. and Havelund K. (2005). Rewriting-Based Techniques for Run-time Verification. Automated Software Engineering, Vol. 12, No. 2, 151–197.

Sha L. (2001). Using Simplicity to Control Complexity. IEEE Software, Vol. 18, No. 4, 20–28.

Sha L. and Meseguer J. (2008). Design of Complex Cyber Physical Systems with Formalized Architectural Patterns, Wirsing M. et al. (Eds.): Software-Intensive Systems, LNCS 5380, Springer-Verlag Berlin Heidelberg, 92–100.

Schierman J. D. et al. (2008). Run-Time Verification and Validation for Safety-Critical Flight Control Systems, AIAA Guidance, Navigation and Control Conference and Exhibit, August 2008, Honolulu, Hawaii. 792–812.

Schneider, D., Trapp, M. (2011). A Safety Engineering Framework for Open Adaptive Systems. In: Proceedings of Fifth IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2011), 89–98.

Seiger R., Keller C., Niebling F. and Schlegel T. (2015). Modelling Complex and Flexible Processes for Smart Cyber-Physical Environments; Journal of Computational Science, Vol. 10, Sept. 2015, 137–148.

Souyris J., Wiels V., Delmas D. and Delseny H. (2009). Formal Verification of Avionics Software Products, A. Cavalcanti and D. Dams (Eds.): FM 2009, LNCS 5850, Springer-Verlag. 532–546.

Sollazzo A., Morani G. and Giovannini A. (2010). An Adaptive Fault-Tolerant FCS for a Large Tranport Aircraft, Edwards C., Lombaerts T., Smaili H. (eds.), Fault Tolerant Flight Control: A Benchmark Challenge, Springer, 272–291.

Tavčar J. and Horváth I. (2018). A Review of the Principles Of Designing Smart Cyber-Physical Systems for Run-Time Adaptation: Learned Lessons and Open Issues, IEEE Transactions on systems, Man, and Cybernetics, Systems. 2018, 1–14.

Teferra M.N. (2017). ISO 14971 - Medical Device Risk Management Standard, International Journal of Latest Research in Engineering and Technology, Vol. 3, No. 3, 83–89.

Toben T., Eilers S., Kuka C., Schweigert S., Winkelmann H. and Ruehrup S. (2012). Safe Autonomous Transport Vehicles in Heterogeneous Outdoor Environments, SARS 2011 and MLSC 2011, Vienna, Austria. 61–75.

Urmson C. et al. (2008). Autonomous Driving in Urban Environments: Boss and the Urban Challenge, Journal of Field Robotics, Vol. 25, No. 8, 425–466.

Xie G., Zeng G., Li Z., Li R. and Li K. (2017). Adaptive Dynamic Scheduling on Multifunctional Mixed-Criticality Automotive Cyber-Physical Systems; IEEE transactions on vehicular tecnology, Vol. 66, No. 8, August 2017, 6676–6692.

Wei S., Zou Y., Zhang T., Zhang X. and Wang W. (2018). Design and Experimental Validation of a Cooperative Adaptive Cruise Control System Based on Supervised Reinforcement Learning, Applied sciences, Vol. 8, No. 7, 1014, 1–21.

Weyns D. et al. (2013). On Patterns for Decentralized Control in Self-Adaptive Systems. In: de Lemos R., Giese H., Müller H.A., Shaw M. (eds) Software Engineering for Self-Adaptive Systems II. Lecture Notes in Computer Science, vol 7475. Springer, Berlin, Heidelberg, 76–107.

Zavala E., Franch X., Marco J., Knauss A., and Damian D. (2018). SACRE: Supporting contextual requirements' adaptation in modern self-adaptive systems in the presence of uncertainty at runtime. Expert Systems with Applications, Vol. 98, 166-188.

Zheng B., Lin C.W., Yu H.,. Liang H and Zhu Q. (2016). "CONVINCE: A Cross-Layer Modeling, Exploration and Validation Framework for Next-Generation Connected Vehicles," in Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD), Nov. 2016, 1–8.

Zhu Q. and Sangiovanni-Vincentelli (2018). Codesign Methodologies and Tools for Cyber–Physical Systems; Proceedings of the IEEE | Vol. 106, No. 9, September 2018, 1484–1500.

Zheng B., Deng P., Anguluri R., Zhu Q. and Pasqualetti F. (2016). Cross-layer Codesign for Secure Cyber-Physical Systems, IEEE Trans. Comput.-Aided Design Integr. Circuits Syst., Vol. 35, No. 5, 699–711.

Zheng X., Julien C., Kim M., and Khurshid S. (2017). Perceptions on the state of the art in verification and validation in cyber-physical systems, IEEE Systems Journal, Vol. 11, No. 4, 2614-2627.

Zikria Y.B., Yu H., Afzal M.K., Rehmani M.H. and Hahmd O. (2018). Internet of Things (IoT): Operating System, Applications and Protocols Design, and Validation Techniques, Future Generation Computer Systems, Vol. 88, Nov. 2018, 699–706.